CLAIMS

1. A computer-implemented method for analyzing trace information generated during execution of multiple threads of a software program on a first computer, the first computer having multiple processors that each have multiple protection domains that are each able to execute at least one of the multiple threads, each processor having a counter indicating a number of instruction holes during which an instruction is not executed by the processor, each protection domain having a counter indicating a number of instructions issued in the protection domain by all executing threads, the method comprising:

receiving an indication of trace information reflecting a series of events that occurred during the execution, each event associated with execution of one of the multiple threads by one of the protection domains of one of the processors and each event having associated values in the trace information of variables maintained by the executing software program, by the one protection domain, and/or by the one processor;

for each of a plurality of periods of time during which the execution was occurring,

determining from the trace information a number of instructions executed for the software program during the period of time by

identifying multiple protection domains that each executed at least one of the multiple threads during at least a portion of the period of time;

for each of the identified protection domains,

determining a change in the value of the issued instructions counter of the protection domain during the period of time;

determining if all of the instructions issued in the protection domain during the period of time were for one of the multiple threads;

when it is determined that all of the instructions issued in the protection domain during the period of time were for one of the multiple threads,

calculating a value for the number of instructions executed for the software program during the period of time by the protection domain to be the determined change; and

when it is determined that all of the instructions issued in the protection domain during the period of time were not for one of the multiple threads, calculating a value for the number of instructions executed for the software program during the period of time by the protection domain to be a portion of the determined change that corresponds to a portion of the period of time during which at least one thread for the software program was executing in the protection domain; and

determining the number of instructions executed for the software program during the period of time to be a sum of the calculated values for each of the identified protection domains; and

determining from the trace information a number of instruction slots available for execution of the instructions of software program during the period of time by

identifying processors that each executed at least one of the multiple threads during the period of time;

for each of the identified processors,

determining a change in the value of the instruction holes counter of the processor during the period of time; and

if all of the instruction holes that occurred during the period of time were attributable to the software program, calculating a value for the number of instruction holes for the processor that are attributable to the software program during the period of time to be the determined change in the value of the instruction holes counter;

calculating a value for the number of instruction holes that are attributable to the software program during the period of time by all of the identified processors to be a sum of the calculated values for each of the identified processors; and

determining the number of instruction slots available for

execution of the instructions of software program during the period of time to be a sum of the determined number of instructions executed for the software program during the period of time and of the calculated value for the number of instruction holes that are attributable to the software program during the period of time; and

presenting to a user an indication of the determined number of executed instructions for each of the periods of time and an indication of the determined number of available instruction slots for each of the periods of time.

2. The method of claim 1 wherein only one software program can execute in a protection domain at any point in time, and wherein when it is determined that all of the instructions issued in a protection domain during a period of time were not for one of the multiple threads, the calculating of a value for the number of instructions executed for the software program during the period of time by the protection domain includes:

determining from the trace information at least one swap event that occurred in the protection domain during the period of time such that the software program is swapped into the protection domain so as to commence execution of the software program or such that the software program is swapped out of the protection domain so as to suspend execution of the software program;

retrieving for each of the determined swap events an associated value in the trace information of the issued instructions counter of the protection domain; and

using the retrieved associated values to calculate the value for the number of instructions executed for the software program during the period of time by the protection domain to include only increments to the issued instructions counter that occurred while the software program is swapped into the protection domain.

3. The method of claim 1 wherein, for at least one of the identified protection domains for at least one of the periods of time, there are no variable values in

the trace information indicating a value for the issued instructions counter of that protection domain at an end of that period of time, and wherein the determining of a second value for the issued instructions counter of that protection domain at the end of that period of time includes estimating the second value based on an extrapolation between earlier and later values for that issued instructions counter.

4. The method of claim 1 wherein, for at least one of the identified processors for at least one of the periods of time, there are no variable values in the trace information indicating a value for the instruction holes counter of that processor at an end of that period of time, and wherein the determining of a second value for the instruction holes counter of that processor at the end of that period of time includes estimating the second value based on an extrapolation between earlier and later values for that instruction holes counter.

5. The method of claim 1 wherein, for at least one of the identified protection domains for at least one of the periods of time, no event occurred during the execution of the software program for that protection domain for that period of time, and including estimating the first and second values for the issued instructions counter of that protection domain.

6. The method of claim 1 wherein, for at least one of the identified processors for at least one period of time, at least one other software program is executing during that period of time, and including, for each of the at least one identified processors:

when it is determined that all of the instruction holes that occurred during that period of time were not attributable to the software program,

calculating a value for the number of instruction holes for that processor that are attributable to the at least one other software programs during that

period of time; and

calculating a value for the number of instruction holes for that processor that are attributable to the software program during that period of time to be a difference of a total number of instruction holes for that processor during that period of time and the calculated value for the number of instruction holes for that processor that are attributable to the at least one other software programs.

7. The method of claim 6 wherein the at least one other software programs include only an operating system program, and wherein the calculated value for the number of instruction holes for each processor that are attributable to the operating system are zero.

8. The method of claim 1 wherein at least one of processors performing the execution has multiple streams performing the execution such that each of the multiple streams executes at least one of the threads, and including displaying information about the streams.

9. The method of claim 1 wherein the presenting to the user of the indication of the determined number of executed instructions for each of the periods of time and of the indication of the determined number of available instruction slots for each of the periods of time includes displaying a graph including the indications.

10. The method of claim 9 wherein the displayed indication of the determined number of available instruction slots for each period of time includes a displayed indication of the calculated number of instruction holes that are attributable to the software program during the period of time, with the calculated number of instruction holes displayed in such a manner that a user can visually aggregate the displayed indication of the determined number of executed instructions for that period of time with

the displayed indication of calculated number of instruction holes for that period of time.

11. The method of claim 9 wherein the displayed graph includes a time-based axis, and wherein the displayed indications of the determined number of executed instructions and the determined number of available instruction slots for each of the periods of time are points on the graph.

12. The method of claim 11 wherein the displayed graph includes an origin with at least two axes, and including, after the displaying of the indications of the determined number of executed instructions and of the determined number of available instruction slots, redefining at least one of the axes based on a new indicated displayed location.

13. The method of claim 1 including, for at least one of the periods of time, presenting an indication of a logical code block of the software program that was executing during that period of time.

14. The method of claim 13 wherein the presented indication of the logical code block is a name of the logical code block.

15. The method of claim 13 wherein the presented indication of the logical code block is source code of the logical code block.

16. The method of claim 13 wherein the logical code block is a function.

17. The method of claim 1 including presenting at least some of the variable values from the trace information in a tabular format.

18. The method of claim 1 wherein the number of processors identified during each of the periods of time is greater than one, and wherein the determined number of available instruction slots for each of the periods of time is the identified number of processors for that period of time.

19. The method of claim 1 wherein a first number of processors identified during a first period of time is distinct from a second number of processors identified during a second period of time.

20. The method of claim 1 wherein the identified number of processors for at least one of the periods of time is greater than one, and wherein information for each of the processors is aggregated during the presenting of information for those periods of time.

21. The method of claim 20 including normalizing the information for each of the processors prior to the presenting of the information by determining a value for the instruction holes counter of each of the processors at a beginning of the execution of the software program and by subtracting the determined value for each instruction hole counter from each later value of that instruction hole counter.

22. The method of claim 1 wherein information to be presented is determined based on a specification provided by a user.

23. The method of claim 22 wherein the specification is used at the time of the presenting to determine the information to be presented.

24. The method of claim 1 including presenting information about

memory references performed during at least one of the periods of time.

25. The method of claim 1 including presenting information about FLOPS performed during at least one of the periods of time.

26. The method of claim 1 including presenting information about a number of the threads in existence during at least one of the periods of time.

27. The method of claim 1 including presenting information about a number of the threads that are blocked during at least one of the periods of time.

28. The method of claim 1 including presenting information about a number of the threads that are ready for execution during at least one of the periods of time.

29. The method of claim 1 including presenting information about contention for locks during at least one of the periods of time.

30. The method of claim 1 wherein the presenting of information is in response to a request by a user.

31. The method of claim 1 wherein the presenting of information is performed automatically without user intervention.

32. The method of claim 1 including generating the indicated trace information.

33. The method of claim 1 wherein the software program is an

executable version of a source code program such that execution of the executable version will generate the indicated trace information.

34. The method of claim 33 including generating the executable version by compiling the source code program such that a group of instructions is added to the source code program at a location specified by the user, the added instructions when executed to generate trace information for a type of event specified by the user.

35. The method of claim 33 including generating the executable version by compiling the source code program such that groups of instructions are added to the source code program at a plurality of automatically identified locations of the software, the added instructions when executed to generate trace information of a specified type.

36. The method of claim 35 wherein the automatically identified locations include beginnings of functions having more lines than a threshold.

37. The method of claim 35 wherein the automatically identified locations include ends of functions having more lines than a threshold.

38. The method of claim 35 wherein the automatically identified locations include locations where a compiler adds instructions related to parallelizing the execution of the executable version.

39. The method of claim 38 wherein the instructions related to parallelizing the execution of the executable version are instructions related to a fork.

40. The method of claim 38 wherein the instructions related to parallelizing the execution of the executable version are instructions related to a join.

41. The method of claim 34 wherein, for at least some of the groups of instructions to be added to the executable program to generate trace information of a specified type, an additional group of instructions is added to the source code program that when executed create a descriptor object for that group of instructions.

42. The method of claim 41 wherein the additional groups of instructions are added to the source code program in such a manner that the additional groups of instructions will be executed before any of the groups of instructions.

43. The method of claim 1 including generating the indicated trace information by executing the executable version.

44. The method of claim 43 wherein, during the executing of the executable version, execution of each of the added groups of instructions adds current values of one or more of the variables to the generated trace information.

45. The method of claim 43 wherein, during the executing of the executable version and before the execution of an added group of instructions, an additional group of added instructions is executed and creates a descriptor object for that added group of instructions.

46. The method of claims 43 wherein information is added to the generated indicated trace information by an executing program other than the software program.

47. The method of claim 46 wherein the other executing program is an operating system.

48. The method of claim 46 wherein the other executing program is a background daemon.

49. The method of claim 1 including analyzing the trace information to extract execution information corresponding to the events that occurred during the execution.

50. The method of claim 49 wherein the analyzing of the trace information includes using a description of specified types of events to identify groups of trace information each reflecting execution of a group of added instructions.

51. The method of claim 50 wherein the analyzing of the trace information includes, when information for an event in the trace information is separated in multiple non-contiguous portions, creating a mapping to assist in retrieving the information for the event.

52. The method of claim 51 including using the identified groups of trace information and the created mapping to extract current values for execution information.

53. The method of claim 49 wherein the analyzing of the trace information includes modifying current values of events so as to be normalized with respect to beginning of the execution.

54. The method of claim 49 wherein the analyzing of the trace information includes modifying current values of events so that the values reflect only the execution of the multiple threads.

55. The method of claim 49 including using information retrieved from created descriptor objects in the trace information.

56. A computer system for analyzing execution of multiple threads of a software program on a first computer, the first computer having multiple processors that each have multiple protection domains that are each able to execute at least one of the multiple threads, each processor having a counter indicating a number of instruction slot holes during which an instruction is not executed by the processor, each protection domain having a counter indicating a number of instructions executed in the protection domain by all executing threads, comprising:

means for analyzing that

receives an indication of trace information reflecting a series of events that occurred during the execution, each event associated with execution of one of the multiple threads by one of the protection domains of one of the processors and each event having associated values in the trace information of variables maintained by the executing software program, by the one protection domain, and/or by the one processor;

for each of a plurality of periods of time during which the execution was occurring, determines from the trace information a number of instructions executed for the software program during the period of time by

identifying multiple protection domains that each executed at least one of the multiple threads during at least a portion of the period of time;

calculating for each of the identified protection domains a value for the number of instructions executed for the software program during the period of time by the protection domain to be a portion of a change in the executed instructions counter of the protection domain during the period of time such that the portion of the change corresponds to a portion of the period of time during which at least one thread for the software program was executing in the protection domain; and

determining the number of instructions executed for the software program during the period of time to be a sum of the calculated values for each of the identified protection domains; and

for each of the plurality of periods of time during which the execution was occurring, determines from the trace information a number of instruction slots available for execution of the instructions of software program during the period of time by

identifying processors that each executed at least one of the multiple threads during the period of time;

calculating for each of the identified processors a value for the number of instruction slot holes for the processor that are attributable to the software program during the period of time to be a portion of a change in the instruction slot holes counter of the processor during the period of time such that other portions of the change corresponding to execution of other software programs by the processor are not included in the portion; and

determining the number of instruction slots available for execution of the instructions of software program during the period of time to be a sum of the determined number of instructions executed for the software program during the period of time and of a sum of the calculated values for each of the identified processors for the period of time; and

means for displaying an indication of the determined number of executed instructions for each of the periods of time and an indication of the determined number of available instruction slots for each of the periods of time.

57. A computing device for analyzing execution of multiple threads of a software program on a first computer, the first computer having multiple processors that each have multiple protection domains that are each able to execute at least one of the multiple threads, each processor having a counter indicating a number of instruction slot

holes during which an instruction is not executed by the processor, each protection domain having a counter indicating a number of instructions executed in the protection domain by all executing threads, comprising:

a trace information receiver component capable of receiving an indication of trace information reflecting a series of events that occurred during the execution, each event associated with execution of one of the multiple threads by one of the protection domains of one of the processors and each event having associated values in the trace information of variables maintained by the executing software program, by the one protection domain, and/or by the one processor;

a trace information analysis component capable of, for each of a plurality of periods of time during which the execution was occurring,

determining from the trace information a number of instructions executed for the software program during the period of time by

identifying multiple protection domains that each executed at least one of the multiple threads during at least a portion of the period of time;

for each of the identified protection domains,

determining a change in the value of the issued instructions counter of the protection domain during the period of time;

determining if all of the instructions issued in the protection domain during the period of time were for one of the multiple threads;

when it is determined that all of the instructions issued in the protection domain during the period of time were for one of the multiple threads, calculating a value for the number of instructions executed for the software program during the period of time by the protection domain to be the determined change; and

when it is determined that all of the instructions issued in the protection domain during the period of time were not for one of the multiple threads, calculating a value for the number of instructions executed for the software program during the period of time by the protection domain to be a portion of the

determined change that corresponds to a portion of the period of time during which at least one thread for the software program was executing in the protection domain; and

determining the number of instructions executed for the software program during the period of time to be a sum of the calculated values for each of the identified protection domains; and

determining from the trace information a number of instruction slots available for execution of the instructions of software program during the period of time by

identifying processors that each executed at least one of the multiple threads during the period of time;

for each of the identified processors,

determining a change in the value of the instruction holes counter of the processor during the period of time; and

if all of the instruction holes that occurred during the period of time were attributable to the software program, calculating a value for the number of instruction holes for the processor that are attributable to the software program during the period of time to be the determined change in the value of the instruction holes counter;

calculating a value for the number of instruction holes that are attributable to the software program during the period of time by all of the identified processors to be a sum of the calculated values for each of the identified processors; and

determining the number of instruction slots available for execution of the instructions of software program during the period of time to be a sum of the determined number of instructions executed for the software program during the period of time and of the calculated value for the number of instruction holes that are attributable to the software program during the period of time; and

a presentation component capable of presenting to a user an indication of the determined number of executed instructions for each of the periods of time and an

indication of the determined number of available instruction slots for each of the periods of time.

58. The computing device of claim 57 wherein the trace information receiver component, trace information analysis component and presentation component are executing in memory of the computing device.

59. A computer-readable medium whose contents cause a computing device to analyze execution of multiple threads of a software program on a first computer, the first computer having multiple processors that each have multiple protection domains that are each able to execute at least one of the multiple threads, each processor having a counter indicating a number of instruction holes during which an instruction is not executed by the processor, each protection domain having a counter indicating a number of instructions issued in the protection domain by all executing threads, by:

receiving an indication of trace information reflecting a series of events that occurred during the execution, each event associated with execution of one of the multiple threads by one of the protection domains of one of the processors and each event having associated values in the trace information of variables maintained by the executing software program, by the one protection domain, and/or by the one processor;

for each of a plurality of periods of time during which the execution was occurring,

determining from the trace information a number of instructions executed for the software program during the period of time by

identifying multiple protection domains that each executed at least one of the multiple threads during at least a portion of the period of time;

for each of the identified protection domains,

determining a change in the value of the issued

instructions counter of the protection domain during the period of time;

determining if all of the instructions issued in the protection domain during the period of time were for one of the multiple threads;

when it is determined that all of the instructions issued in the protection domain during the period of time were for one of the multiple threads, calculating a value for the number of instructions executed for the software program during the period of time by the protection domain to be the determined change; and

when it is determined that all of the instructions issued in the protection domain during the period of time were not for one of the multiple threads, calculating a value for the number of instructions executed for the software program during the period of time by the protection domain to be a portion of the determined change that corresponds to a portion of the period of time during which at least one thread for the software program was executing in the protection domain; and

determining the number of instructions executed for the software program during the period of time to be a sum of the calculated values for each of the identified protection domains; and

determining from the trace information a number of instruction slots available for execution of the instructions of software program during the period of time by

identifying processors that each executed at least one of the multiple threads during the period of time;

for each of the identified processors,

determining a change in the value of the instruction holes counter of the processor during the period of time; and

if all of the instruction holes that occurred during the period of time were attributable to the software program, calculating a value for the number of instruction holes for the processor that are attributable to the software program during the period of time to be the determined change in the value of the instruction holes

counter;

calculating a value for the number of instruction holes that are attributable to the software program during the period of time by all of the identified processors to be a sum of the calculated values for each of the identified processors; and

determining the number of instruction slots available for execution of the instructions of software program during the period of time to be a sum of the determined number of instructions executed for the software program during the period of time and of the calculated value for the number of instruction holes that are attributable to the software program during the period of time; and

presenting to a user an indication of the determined number of executed instructions for each of the periods of time and an indication of the determined number of available instruction slots for each of the periods of time.

60. The computer-readable medium of claim 59 wherein the computer-readable medium is a memory of a computer.

61. The computer-readable medium of claim 59 wherein the computer-readable medium is a transmission medium containing generated data that includes the contents.

62. A method for generating trace information reflecting a series of events that occurred during execution of multiple software threads on a first computer, the method comprising:

receiving an indication of a software program for which trace information is to be generated;

receiving indications from a user of one or more locations within the software program at which trace information is to be generated and of one or more types of event each indicating distinct types of trace information; and

automatically producing an executable version corresponding to the software program such that when executed the produced executable version will generate trace information corresponding to multiple events of the types indicated by the user, the producing of the executable version including adding multiple groups of instructions to the software program at the locations specified by the user, each of the added groups of instructions corresponding to one of the types of events indicated by the user such that when executed that added group of instructions will generate the distinct types of trace information for that type of event, the produced executable version having one or more portions which can be executed in parallel with multiple software threads,
so that execution of the produced executable version will generate trace information reflecting a series of events that occurred during execution.

63. The method of claim 62 including executing the produced executable version to generate the trace information.

64. The method of claim 62 wherein the software program is a source code program, and wherein the producing of the executable version includes compiling the source code program.

65. The method of claim 64 wherein the multiple groups of instructions are added to the software program before the compiling.

66. The method of claim 62 wherein the producing of the executable version includes adding groups of instructions at a plurality of automatically identified locations of the software, the added groups of instructions to each generate trace information of a specified type when executed.

67. The method of claim 66 wherein the automatically identified

locations include beginnings of functions having more lines than a threshold.

68. The method of claim 66 wherein the automatically identified locations include ends of functions having more lines than a threshold.

69. The method of claim 66 wherein the automatically identified locations include locations where a compiler adds instructions related to parallelizing the execution of the executable version.

70. The method of claim 69 wherein the instructions related to parallelizing the execution of the executable version are instructions related to a fork.

71. The method of claim 69 wherein the instructions related to parallelizing the execution of the executable version are instructions related to a join.

72. The method of claim 62 wherein, for at least some of the groups of instructions added to the executable program, additional instructions are added to the executable program that when executed creates a descriptor object for that group of instructions.

73. The method of claim 72 wherein the additional groups of instructions are added in such a manner as to be executed before any of the added groups of instructions.

74. The method of claim 62 wherein the first computer has multiple processors that each have multiple protection domains each able to execute at least one of the multiple threads.

75. The method of claim 62 wherein the first computer provides multiple sources of information related to hardware aspects of the first computer that vary with execution of programs on the first computer, and wherein the distinct types of trace information for some of the types of events includes at least one of the sources of information related to the hardware aspects.

76. The method of claim 62 wherein the first computer provides multiple sources of information related to software properties of the first computer that vary with execution of programs on the first computer, and wherein the distinct types of trace information for some of the types of events includes at least one of the sources of information related to the software properties.

77. A computer-readable medium containing instructions that when executed cause a computing device to generate trace information reflecting a series of events that occurred during execution of multiple software threads on a first computer, by:

receiving an indication of a software program for which trace information is to be generated;

receiving indications of one or more locations within the software program at which trace information is to be generated and of one or more types of event each indicating distinct types of trace information to be generated; and

automatically producing an executable version corresponding to the software program such that when executed the produced executable version will generate trace information corresponding to multiple events of the indicated types, the producing of the executable version including adding multiple groups of instructions to the software program at the locations specified by the user, each of the added groups of instructions corresponding to one of the indicated types of events such that when executed that added

group of instructions will generate the distinct types of trace information for that type of event, the produced executable version having one or more portions which can be executed in parallel with multiple software threads.

78. A computing device for generating trace information reflecting a series of events that occurred during execution of multiple software threads on a first computer, comprising:

an input receiver component capable of receiving an indication of a software program for which trace information is to be generated, and of receiving indications from a user of one or more locations within the software program at which trace information is to be generated and of one or more types of event each indicating distinct types of trace information; and

an executable generator component capable of producing an executable version corresponding to the software program such that when executed the produced executable version will generate trace information corresponding to multiple events of the types indicated by the user, the producing of the executable version including adding multiple groups of instructions to the software program at the locations specified by the user, each of the added groups of instructions corresponding to one of the types of events indicated by the user such that when executed that added group of instructions will generate the distinct types of trace information for that type of event, the produced executable version having one or more portions which can be executed in parallel with multiple software threads.

79. A method for generating trace information reflecting a series of events that occurred during execution of multiple software threads on a first computer, the method comprising:

receiving an indication of an executable software program that when executed will generate trace information corresponding to multiple events of specified

types, the executable software program including multiple groups of instructions added at specified locations and each corresponding to one of the specified types of events; and

generating the trace information by executing the executable software program using multiple software threads on the first computer in such a manner that each of the added group of instructions are executed at least once, each execution of an added group of instructions corresponding to a specified type of event generating trace information of the type for that type of event.

80. The method of claim 79 including, before the generating of the trace information by the executing of the executable software, producing the executable software program from software source code in such a manner that the multiple groups of instructions are added at the specified locations.

81. The method of claim 79 wherein the specifications of the locations and of the types of events for the multiple added groups of instructions are received from a user.

82. The method of claim 79 wherein, during the executing of the executable software program, the execution of each of the added groups of instructions adds current values of one or more variables to the generated trace information, the one or more variables maintained by the executing software program and/or by the first computer.

83. The method of claim 79 wherein the first computer has multiple processors that each have multiple protection domains each able to execute at least one of the multiple threads.

84. The method of claim 79 wherein, during the executing of the

executable software program and before the execution of an added group of instructions, an additional group of added instructions is executed and creates a descriptor object for that added group of instructions.

85. The method of claims 79 wherein information is added to the generated indicated trace information by an executing program other than the executable software program.

86. The method of claim 85 wherein the other executing program is an operating system.

87. The method of claim 85 wherein the other executing program is a background daemon.

88. A computer-readable medium whose contents cause a computing device to generate trace information reflecting a series of events that occurred during execution of multiple software threads on a first computer, by:

receiving an indication of an executable software program that when executed will generate trace information corresponding to multiple events of specified types, the executable software program including multiple groups of instructions added at specified locations and each corresponding to one of the specified types of events; and

generating the trace information by executing the executable software program using multiple software threads on the first computer in such a manner that each of the added group of instructions are executed at least once, each execution of an added group of instructions corresponding to a specified type of event generating trace information of the type for that type of event.

89. A computing device for generating trace information reflecting a

series of events that occurred during execution of multiple software threads on a first computer, comprising:

an input receiver component capable of receiving an indication of an executable software program that when executed will generate trace information corresponding to multiple events of specified types, the executable software program including multiple groups of instructions added at specified locations and each corresponding to one of the specified types of events; and

a trace information generator component capable of generating the trace information by executing the executable software program using multiple software threads on the first computer in such a manner that each of the added group of instructions are executed at least once, each execution of an added group of instructions corresponding to a specified type of event generating trace information of the type for that type of event.

90. A method for analyzing trace information generated during execution of multiple threads of a software program on a first computer, the generated trace information reflecting a series of events that occurred during the execution, the method comprising:

receiving an indication of trace information generated during execution of an executable software program using multiple software threads on the first computer, the executable software program including multiple groups of instructions each corresponding to a specified type of event, the execution such that each of the multiple group of instructions are executed at least once and such that each execution of an added group of instructions generates trace information of a type corresponding to the specified type of event for that added group of instructions; and

analyzing the generated trace information to extract execution information corresponding to the events that occurred during the execution

91. The method of claim 90 including producing the executable software program from software source code in such a manner that the multiple groups of instructions are added at the specified locations.

92. The method of claim 90 including generating the indicated trace information by executing the executable software program using multiple software threads on the first computer.

93. The method of claim 90 wherein, during the executing of the executable software program, the execution of each of the added groups of instructions adds current values of one or more variables to the generated trace information, the one or more variables maintained by the executing software program and/or by the first computer.

94. The method of claim 90 wherein the first computer has multiple processors that each have multiple protection domains each able to execute at least one of the multiple threads.

95. The method of claim 90 wherein the analyzing of the generated trace information includes using a description of the specified types of events to identify groups of trace information each reflecting execution of a group of added instructions corresponding to those specified types of events.

96. The method of claim 95 wherein the analyzing of the trace information includes, when information for an event in the generated trace information is separated in multiple non-contiguous portions, creating a mapping to assist in retrieving the information for the event.

97. The method of claim 96 including using the identified groups of trace information and the created mapping to extract current values for execution information.

98. The method of claim 90 wherein the analyzing of the trace information includes modifying current values of events so as to be normalized with respect to beginning of the execution.

99. The method of claim 90 wherein the analyzing of the trace information includes modifying current values of events so that the values reflect only the execution of the multiple threads.

100. The method of claim 90 including using information retrieved from created descriptor objects in the generated trace information.

101. A computer-readable medium whose contents cause a computing device to analyze trace information generated during execution of multiple threads of a software program on a first computer, the generated trace information reflecting a series of events that occurred during the execution, by:

receiving an indication of trace information generated during execution of an executable software program using multiple software threads on the first computer, the executable software program including multiple groups of instructions each corresponding to a specified type of event, the execution such that each of the multiple group of instructions are executed at least once and such that each execution of an added group of instructions generates trace information of a type corresponding to the specified type of event for that added group of instructions; and

analyzing the generated trace information to extract execution information

corresponding to the events that occurred during the execution

102. A computing device for analyzing trace information generated during execution of multiple threads of a software program on a first computer, the generated trace information reflecting a series of events that occurred during the execution, comprising:

an input receiver component capable of receiving an indication of trace information generated during execution of an executable software program using multiple software threads on the first computer, the executable software program including multiple groups of instructions each corresponding to a specified type of event, the execution such that each of the multiple group of instructions are executed at least once and such that each execution of an added group of instructions generates trace information of a type corresponding to the specified type of event for that added group of instructions; and

a trace information analysis component capable of analyzing the generated trace information to extract execution information corresponding to the events that occurred during the execution.

103. A method for analyzing trace information generated during execution of multiple threads of a software program on a first computer, the generated trace information reflecting a series of events that occurred during the execution, the method comprising:

receiving an indication of execution information extracted from trace information generated during execution of an executable software program using multiple software threads on the first computer, the execution information corresponding to events that occurred during the execution, the executable software program including multiple groups of instructions each corresponding to a specified type of event, the execution such that each of the multiple group of instructions are executed at least once and such that

each execution of an added group of instructions generates trace information of a type corresponding to the specified type of event for that added group of instructions; and

for each of multiple periods of time, presenting to a user an indication of the extracted execution information for that period of time.

104. The method of claim 103 including analyzing the generated trace information to extract execution information.

105. The method of claim 103 wherein the presenting to the user includes displaying a graph including the indications.

106. The method of claim 103 wherein the indicated extracted execution information includes an indication of a determined number of executed instructions for each of the periods of time and an indication of a determined number of available instruction slots for each of the periods of time.

107. The method of claim 106 wherein the displayed indication of the determined number of available instruction slots for each period of time includes a displayed indication of a calculated number of instruction holes that are attributable to the executing software program during the period of time, with the calculated number of instruction holes displayed in such a manner that a user can visually aggregate the displayed indication of the determined number of executed instructions for that period of time with the displayed indication of calculated number of instruction holes for that period of time.

108. The method of claim 106 wherein the displayed graph includes a time-based axis, and wherein the displayed indications of the determined number of executed instructions and the determined number of available instruction slots for each of

the periods of time are points on the graph.

109. The method of claim 108 wherein the displayed graph includes an origin with at least two axes, and including, after the displaying of the indications of the determined number of executed instructions and of the determined number of available instruction slots, redefining at least one of the axes based on a new indicated displayed location.

110. The method of claim 103 including, for at least one of the periods of time, presenting an indication of a logical code block of the executable software program that was executing during that period of time.

111. The method of claim 110 wherein the presented indication of the logical code block is a name of the logical code block.

112. The method of claim 110 wherein the presented indication of the logical code block is source code of the logical code block.

113. The method of claim 110 wherein the logical code block is a function.

114. The method of claim 103 wherein the extracted execution information includes values of one or more variables maintained by the executing software program and/or by the first computer, and including presenting at least some of the included variable values in a tabular format.

115. The method of claim 103 wherein a number of processors of the first computer that are identified during each of the periods of time is greater than one, and

wherein a determined number of available instruction slots for each of the periods of time is the identified number of processors for that period of time.

116. The method of claim 103 wherein an identified number of processors of the first computer for at least one of the periods of time is greater than one, and wherein information for each of the processors is aggregated during the presenting of information for those periods of time.

117. The method of claim 116 including normalizing the information for each of the processors prior to the presenting of the information by determining a value for an instruction holes counter of each of the processors at a beginning of the execution of the software program and by subtracting the determined value for each instruction hole counter from each later value of that instruction hole counter.

118. The method of claim 103 wherein information to be presented is determined based on a specification provided by a user.

119. The method of claim 118 wherein the specification is used at the time of the presenting to determine the information to be presented.

120. The method of claim 103 including presenting information about memory references performed during at least one of the periods of time.

121. The method of claim 103 including presenting information about FLOPS performed during at least one of the periods of time.

122. The method of claim 103 including presenting information about a number of the threads in existence during at least one of the periods of time.

123. The method of claim 103 including presenting information about a number of the threads that are blocked during at least one of the periods of time.

124. The method of claim 103 including presenting information about a number of the threads that are ready for execution during at least one of the periods of time.

125. The method of claim 103 including presenting information about contention for locks during at least one of the periods of time.

126. The method of claim 103 wherein the presenting of information is in response to a request by a user.

127. The method of claim 103 wherein the presenting of information is performed automatically without user intervention.

128. A computer-readable medium whose contents cause a computing device to analyze trace information generated during execution of multiple threads of a software program on a first computer, the generated trace information reflecting a series of events that occurred during the execution, by:

receiving an indication of execution information extracted from trace information generated during execution of an executable software program using multiple software threads on the first computer, the execution information corresponding to events that occurred during the execution, the executable software program including multiple groups of instructions each corresponding to a specified type of event, the execution such that each of the multiple group of instructions are executed at least once and such that each execution of an added group of instructions generates trace information of a type

corresponding to the specified type of event for that added group of instructions; and

for each of multiple periods of time, presenting to a user an indication of the extracted execution information for that period of time.

129. A computing device for analyzing trace information generated during execution of multiple threads of a software program on a first computer, the generated trace information reflecting a series of events that occurred during the execution, comprising:

an input receiver component capable of receiving an indication of execution information extracted from trace information generated during execution of an executable software program using multiple software threads on the first computer, the execution information corresponding to events that occurred during the execution, the executable software program including multiple groups of instructions each corresponding to a specified type of event, the execution such that each of the multiple group of instructions are executed at least once and such that each execution of an added group of instructions generates trace information of a type corresponding to the specified type of event for that added group of instructions; and

a trace information presenter component capable of, for each of multiple periods of time, presenting to a user an indication of the extracted execution information for that period of time.